

REMARKS

Claims 22-51 were pending at the time of the final Office Action of April 26, 2004 (hereinafter "Office Action"). By this Amendment, claims 25, 31, 35, and 41-51 are canceled; claims 22, 24, 26-30, 32, 34, and 36-40 are amended; and claims 52-61 are added. A total of 26 claims are therefore presented for reconsideration including claims 22-24, 26-30, 32-34, 36-40, and 52-61. All amendments are fully supported by the specification; hence, no new matter has been entered.

Claims 22-51 stand rejected under 35 U.S.C. §103(a) as being unpatentable over U.S. Patent 5,193,180 (Hastings) in view of U.S. Patent 5,822,590 (Gupta). Applicants respectfully traverse for the reasons set forth below, in which all references to claim elements are made to the list of claims provided above in the section entitled, "IN THE CLAIMS." For the Office Action, line numbers are referenced based on all printed lines, including headings, except for the header.

Claim 22 sets forth a method for generating an object-oriented computer program to access and update persistently stored objects (claim 22, lines 1-2). An initial computer program includes original instructions for accessing objects stored in a computer's main memory (lines 4-5). Instructions are automatically added to the computer program to load the objects from persistent storage into the main memory when the object is not already in main memory (lines 11-14). When the object is loaded, it is modified (lines 12-13). Specifically, object data structures of the object are modified to store persistent data descriptors, and new methods are added to access and use these persistent data descriptors (lines 19-22).

Hastings teaches a system for modifying compiled computer code by inserting statements for monitoring memory accesses for debugging purposes. Hastings does not teach an object-oriented computer programming language or an object-oriented database. The use of the word, "object" in Hastings, is consistently used to refer to compiler output, which is called, "object code." Object code is not the same things as the objects containing data referenced in the present claims.

Gupta teaches a computer program extension that modifies source code to provide a modified source code (col. 5, lines 39-47). The extension provides additional keywords for

providing seamless, unfragmented, persistible, complex object heap space that extends beyond virtual memory (col. 1, lines 46-52). Gupta therefore inserts instructions into source code for keeping track of data loaded from disk. Specifically, Gupta establishes a global data structure called, "dbtable" (col. 3, lines 44-46). Gupta does not mention modifying objects loaded from disk nor storing persistent data descriptors, or any analog thereof, in such objects.

Since neither Hastings nor Gupta teach inserting instructions into a program that modifies objects loaded from persistent storage, Applicants respectfully submit that not all claim limitations set forth in claim 22 are taught or suggested by the prior art. Claim 22 should therefore be allowed.

Claims 23, 23, and 52-55 depend from claim 22 and are allowable for the same reasons as claim 22. In addition, claims 23, 24, and 52-55 further define the invention and further distinguish same from the prior art. For example, claim 24 sets forth that each persistent data descriptor includes a pointer to a next dirty object in a linked list of dirty objects. (See, e.g., claim 24, lines 2-4.) When data in the object is modified or updated, the object becomes "dirty," meaning that it is no longer consistent with the copy in persistent storage, and must be written back to the persistent storage. Dirty objects are added to a linked list of dirty objects which is followed when the objects are to be written back to persistent storage (claim 24, lines 8-10).

Neither Hastings nor Gupta teach using a linked list to identify and store dirty objects to persistent storage. Hastings does not mention objects. Gupta does not mention any method of distinguishing clean objects from dirty objects. Since Hastings and Gupta fail to teach or suggest a method for identifying and storing dirty objects, Applicants respectfully submit that claim 24 is allowable. Furthermore, claim 54, which depends from claim 24, should be allowed for the same reasons as claim 24.

Claim 26 sets forth a method for generating object oriented computer program to access and update persistently stored objects (lines 1-2 thereof). An initial computer program includes original instructions for accessing and updating objects stored in a computer's main memory and committing transactions in which one or more objects may have been updated (lines 4-6). The initial computer program is automatically revised to generate a revised computer program (lines 13-14). In the revised computer program, persistent objects are modified by modifying the data structures thereof to store persistent data descriptors and

adding new methods to the persistent objects allowing access and use of the persistent data descriptors (lines 15-19). Furthermore, new instructions are added to the initial computer program that mark dirty objects which contain new and/or updated data so that the dirty objects can be identified (lines 13, 14, and 20-24). In addition, new instructions are added for storing dirty objects into persistent storage using the persistent data descriptors stored in the object data structures of the persistent objects (lines 25-30).

Neither Hastings nor Gupta teach using persistent data descriptors stored in object data structures of the dirty objects to mark and save dirty objects to persistent storage. Hastings does not mention objects. Gupta does not mention any method of distinguishing clean objects from dirty objects. Since Hastings and Gupta fail to teach or suggest a method for identifying and storing dirty objects, Applicants respectfully submit that claim 26 is allowable. Furthermore, claims 27 and 56, which depend from claim 26, should be allowed for the same reasons as claim 26. In addition, claims 27 and 56 further define and distinguish the invention.

For example, claim 27 sets forth that the persistent data descriptors includes a persistent storage object identifier, and object storing instructions further include instructions for replacing local object references in the respective objects with the persistent storage object identifiers in the corresponding data descriptor before storing the respective objects in persistent storage. (See claim 27, lines 1-6). None of the prior art references teach storing full object identifiers in an object loaded into memory. This claim, along with other dependent claims, should therefore be allowed for reasons distinct from depended-upon claim 26.

Claim 28 sets forth a method of generating object-oriented computer programs for accessing and updating persistently stored objects (lines 1-2). An initial computer program is automatically revised to generate a revised computer program (lines 10-11). The revised computer program modifies the persistent objects and adds supplemental instructions to the initial computer program (lines 11-12). Specifically, the persistent objects are modified to store persistent data descriptors are further modified to add methods to the persistent objects allowing access and use of the persistent data descriptors (lines 23-26). Each persistent data descriptor includes a pointer to a next dirty object (line 27). The supplemental instructions add persistent objects to a linked list of dirty objects using the pointer to a next dirty object in the persistent data descriptor when the object contains new and/or updated data (lines 28-30).


Neither Hastings nor Gupta teach using a linked list to identify dirty objects, and furthermore, neither teach using a pointer in a persistent data descriptor in the object itself to effectuate the linked list. Hastings does not mention objects. Gupta does not mention any method of distinguishing clean objects from dirty objects. Since Hastings and Gupta fail to teach or suggest a method for identifying and storing dirty objects, Applicants respectfully submit that claim 28 is allowable. Furthermore, claims 29 and 30, which depends from claim 28, should be allowed for the same reasons as claim 28. Claims 29 and 30 furthermore contain additional claim elements further defining and distinguishing the invention from prior art.

Claims 32-40 and 57-61 run substantially parallel to claims 22-30 and 52-56 discussed above, the former being directed towards a computer program product. Therefore, claims 32-40 and 57-61 should be allowed for the same reasons discussed above with respect to claims 22-30 and 52-56.

In view of the remarks above, Applicants respectfully submit that the present application is in condition for allowance. A Notice of Allowance is therefore respectfully requested.

If the Examiner has any questions concerning the present amendment, the Examiner is kindly requested to contact the undersigned at (408) 749-6900 x6933. If any other fees are due in connection with filing this amendment, the Commissioner is also authorized to charge Deposit Account No. 50-0805 (Order No. SUNMP504C). A duplicate copy of the transmittal is enclosed for this purpose.

Respectfully submitted,
MARTINE & PENILLA, LLP


Leonard E. Heyman, Esq.
Reg. No. 40,418

710 Lakeway Drive, Suite 170
Sunnyvale, CA 94085
Telephone: (408) 749-6900
Facsimile: (408) 749-6901